ACEEE Int. J. on Network Security , Vol. 4, No. 1, July 2013

A Real Time Framework of Multiobjective Genetic Algorithm for Routing in Mobile Networks

Subarno Banerjee, Rajarshi Poddar, and P. K. Guha Thakurta
Department of Computer Science and Engineering
National Institute of Technology, Durgapur, India
Email: {banerjee.subarno, rajarshi.poddar, parag.nitdgp}@gmail.com

Abstract— **Routing in mobile networks is a multiobjective optimization problem. The problem needs to consider multiple objectives simultaneously such as Quality of Service parameters, delay and cost. This paper uses the NSGA-II multiobjectve genetic algorithm to solve the dynamic shortest path routing problem in mobile networks and proposes a framework for real-time software implementation. Simulations confirm a good quality of solution (route optimality) and a high rate of convergence.**

Index Terms— **Real-time communication, mobile networks, shortest path routing, multiobjective optimization, genetic algorithms**

I. Introduction

Routing has a significant impact on the performance of mobile networks. An efficient routing algorithm should find an optimum path for routing while adhering to Quality of Service (QoS) requirements. Several polynomial time Shortest Path (SP) search algorithms [7], such as Dijkstra's algorithm and Bellman-Ford algorithm, work effectively for fixed infrastructure wired or wireless networks. But, they suffer from high computational complexity for real-time applications in mobile networks with rapidly changing topology and/or network status.

Classically, SP routing problem has been formulated to combinatorial optimization that seeks to find the single best solution in one run. Routing in mobile networks involves simultaneous optimization of multiple QoS parameters such as bandwidth, packet delay, loss, etc. Generally, these objectives compete and conflict with each other. Such competition among conflicting objectives gives rise to a set of optimal solutions instead of a single solution. These set of solutions are known as pareto-optimal solutions and no solution can be judged better than others in terms of all objectives. Genetic algorithms can find multiple optimal solutions in a single run due to their population based approach.

A simpler approach is to convert the multiobjective optimization into a single objective optimization problem by formulating a composite objective function as the weighted sum of the objectives [3]. Due to conflict among individual objectives the quality of solution is degenerated. This approach is investigated in [1]. In a true Multi-Objective Genetic Algorithm (MOGA), multiple objectives are traded-off simultaneously and their interactions do not affect the quality of the solution.

Mobile networks need to support a variety of real-time services such as voice/video call, tele-conferencing, etc. In general, GAs involve a large number of iterations and are therefore not good candidates for real-time applications. Only hardware implementations of GAs are fast enough to execute in time-constrained environments. However, hardware implementations (FPGA chips) have high costs of upgradation.

In this paper, we adopt a MOGA based approach for the dynamic SP routing problem in mobile networks. The problem is formulated as a nonlinear constrained multiobjective optimization, where cost and delay are treated as competing objectives. The elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) [4] employs a diversity-preserving mechanism to yield a set of widely diverse pareto-optimal solutions. Furthermore, we introduce a framework that allows for real-time software implementation of GAs for routing in mobile networks.

The paper is organized as follows. In section II, the dynamic SP routing problem is formulated using MOGAs. Section III presents our real-time framework. The results of simulations are summarised in section IV. Finally in section V, we conclude with a discussion on the advantages and future scope of the proposed model.

II. GA for SP Routing

The underlying topology of mobile cellular networks is represented using the coordinate mapping in [2]. Here each mobile cell *n* is represented by an ordered pair of its coordinate values $(x_{n,}y_n)$. Each unordered pair of adjacent cells is associated with a transmission cost and estimated transmission status.

The transmission costs are specified by the cost matrix $C = [c_{ij}]$,where is cost of transmitting a call from cell *i* to its adjacent cell *j*. The cost values depend on the channel capacity and transmission power; costs are generally higher with higher allocated capacity and at lower level of transmission power.

$$c_{ij} = \begin{cases} cost, & i\ and\ j\ are\ adjacent \\ 0, & i = j \\ \infty, & otherwise \end{cases}$$

The transmission status denotes propagation delay or transmission failure. It is specified with the delay matrix $D = [d_{ij}]$ where d_{ij} is the estimated propagation delay from cell *i* to its adjacent cell *j*.

$$d_{ij} = \begin{cases} delay, & i \text{ and } j \text{ are adjacent} \\ 0, & i = j \\ \infty, & \text{otherwise or in case of failure} \end{cases}$$

The cost and delay matrices are sparse matrices [8] with ∞ entries for all non-adjacent cell pairs. Obviously, all diagonal elements must be 0. Each cell pair has a route indicator denoted by r_{ij} which indicates if $i \rightarrow j$ belongs to the route r

$$r_{ij} = \begin{cases} 1, & i \rightarrow j \text{ belongs to } r \\ 0, & \text{otherwise} \end{cases}$$

The problem is to find a path between the source *s* and destination *d* minimizing cost and delay at the same time. The SP routing problem is formulated as a multiobjective non-linear programming problem as follows:

minimize

$$\sum_{i=s}^{d} \sum_{\substack{j=s \\ j \neq i}}^{d} c_{ij} \times r_{ij} \tag{1}$$

and

minimize

$$\sum_{i=s}^{d} \sum_{\substack{j=s \\ j \neq i}}^{d} d_{ij} \times r_{ij} \tag{2}$$

subject to

$$\sum_{\substack{j=s \\ j \neq i}}^{d} r_{ij} - \sum_{\substack{j=s \\ j \neq i}}^{d} r_{ji} = \begin{cases} 1, & i = s \\ -1, & i = d \\ 0, & otherwise \end{cases} \tag{3}$$

and

$$\sum_{\substack{j=s \\ j \neq i}}^{d} r_{ji} \begin{cases} \leq 1, & i \neq d \\ = 0, & i = d \end{cases} \tag{4}$$

Objectives (1) and (2) tend to minimize total cost and end-to-end delay respectively. The cost and delay objectives compete and conflict with each other. (3) denotes the flow conservation constraint and constraint (4) ensures a loop-free path.

A. Genetic Representation

A chromosome in the proposed GA is a route $r_{(s,d)}$ from source *s* to destination *d* represented as a sequence of nodes–

$\{(x_{n_o}, y_{n_o}), (x_{n_1}, y_{n_1}), (x_{n_2}, y_{n_2}), \dots, (x_{n_{h-1}}, y_{n_{h-1}}), (x_{n_h}, y_{n_h})\}$,

where *h* is the hop count of the route, such that:

$n_0 \equiv s$, $n_h \equiv d$,

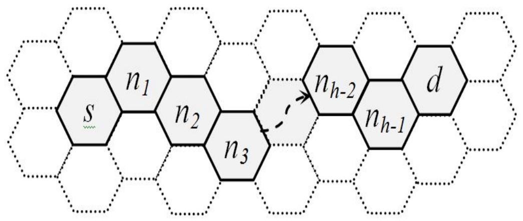
for any i, $1 \leq i \leq h$, $|x_{n_{i-1}} - x_{n_i}| + |y_{n_{i-1}} - y_{n_i}| = 2$,

and $0 < h < N$, where *N* is the number of cells in the network.

A gene of a chromosome is a cell site. The chromosomes have variable length, but this length is upper bounded by the number of cells in the network. While generating chromosomes, the first gene always encodes the source cell and thereafter an adjacent cell is a randomly selected as the next gene. This process is repeated until the destination cell is reached. To ensure loop-free paths, the chosen cell is marked to prevent it from being selected twice and the next cell is chosen only from the unmarked adjacent cells. An example chromosome for a routing path is shown in Fig. 1

B. Population Initialization

A population $P_{(s,d)}$ consists of a subset of all possible



Locus:	1	2	3	4		h-1	h	h+1
Chromosome:	*S*	n_1	n_2	n_3	…	n_{h-2}	n_{h-1}	*d*

Figure 1. Example routing path and its encoded chromosome

routes (chromosomes) between source-destination pair *(s,d)*. The population initialisation considers two issues- population size and the initialisation procedure. Initial population size plays a decisive role. If the population size is too small, it is not likely to find the optimal solutions. However, large population size demands more memory and results in slow convergence. Therefore, an optimum population size is required for a good quality of solution and acceptable convergence rate. We use the generalised population sizing equation [3,6] to determine the initial population size. The initial population is generated in a random manner. It is possible to have defective or duplicate chromosomes in the population; such chromosomes are reinitialised. Random population initialization induces diversity in the population.

C. NSGA-II

The goal of a multiobjective optimization is to find the pareto-optimal front and also maintain population diversity in the set of non-dominated solutions. NSGA-II uses a ranking selection method to emphasize non-dominated solutions and a niching operation to maintain diversity in the population. The main steps that are followed in the algorithm are:

1) Non-dominated Sorting

The basic idea is to find a set of solutions in the population that are non-dominated by the rest of the population. The following steps describe the procedure to find the non-dominated set in a given population *P* of size *N*.

Step 1: Set i =1, and create an empty non-dominated set P'.

Step 2: For a solution $j \in P, j \neq i$, check if solution j dominates solution i. If yes go to step 4.

Step 3: If more solutions are left in P, $j = j + 1$ and go to step 2; otherwise, set $P' = P' \cup \{i\}$.

Step 4: $i = i + 1$. If $i \leq N$, go to step 2; otherwise P' is the non-dominated set.

The set of solutions represent the first non-domination front and are eliminated from further contention. The process continues until the whole population is properly classified into different non-domination fronts. Non-dominated sorting is performed on the combined population (parent + offspring) after each generation to ensure elitism.

1) *Fitness Assignment*

All solutions in the same non-domination front are equally important in terms of their closeness to the pareto-optimal front relative to the current population. Therefore, the same fitness is assigned to all of them based on the number of solutions they dominate and the dominated sets are assigned fitness worse than the worst fitness of any non-dominated solution. Assigning higher fitness to solutions in a better non-dominated set ensures selection pressure towards the pareto-optimal front and thus fast convergence.

2) *Crowding Distance Sorting*

This step preserves diversity among solutions of the same non-domination front. The idea behind niching is: the more solutions are located in the neighbourhood of a certain solution, the more its fitness value is degraded [4]. Thus, a higher fitness is assigned to solutions located on sparsely populated part of the front. Obviously, this yields widely separated solutions within the same non-domination front. NSGA-II uses a density estimation model to define the neighbourhood and sorts the solutions within a non-domination front on the basis of crowding distance. This density-preserving mechanism leads to a uniformly spread-out pareto-optimal front.

D. *Tournament Selection*

Selection essentially improves the average quality of the population by passing high quality chromosomes to the next generation. We employ binary tournament selection procedure. Selection of a chromosome pair is based on their fitness or rank such that they have at least one gene common in addition to the source and destination cells. A selection scheme is characterized by selection pressure [5]. The selection pressure drives the population towards better quality. In each iteration, the *N* existing parents generate *N* new individual offspring. Both parents and offspring compete with each other during non-dominated sorting for inclusion in the next generation.

E. *Crossover*

Crossover generates new paths by exchanging partial paths of selected mating pairs of chromosomes. Single point crossover has been employed previously in [1]. Now we extend the crossover operation to perform multi-point crossover of partial paths. Consider a mating chromosome pair $(r_{(s,d)}{}^{1}, r_{(s,d)}{}^{2})$,

$$r_{(s,d)}{}^{1} = \{(x_{s,}y_s), (x_{n_1{}^1,}y_{n_1{}^1}), \ldots, (x_{n_{h_1-1}{}^1,}y_{n_{h_1-1}{}^1}), (x_{d,}y_d)\}$$

$$r_{(s,d)}{}^{2} = \{(x_{s,}y_s), (x_{n_1{}^2,}y_{n_1{}^2}), \ldots, (x_{n_{h_2-1}{}^2,}y_{n_{h_2-1}{}^2}), (x_{d,}y_d)\}$$

The common gene(s) that appear in the same order in both chromosomes are the potential crossover points. Let there be *m* crossover points at $\{c_1, c_2, \ldots, c_m\}$; $m < h_1$ and $m < h_2$. Multi-point crossover is performed as follows:

Step 1: Split $r_{(s,d)}{}^{1}$ into m+1 parts –

$r_{(s,d)}{}^{1}{}_{1} = \{(x_{s,}y_s), (x_{n_1{}^1,}y_{n_1{}^1}), \ldots, (x_{c_1,}y_{c_1})\}$,

$r_{(s,d)}{}^{1}{}_{2} = \{(x_{c_1,}y_{c_1}), (x_{n\_{}^1,}y_{n\_{}^1}), \ldots, (x_{c_2,}y_{c_2})\}$,

Step 2: Split $r_{(s,d)}{}^{2}$ into m+1 parts –

$r_{(s,d)}{}^{2}{}_{1} = \{(x_{s,}y_s), (x_{n_1{}^2,}y_{n_1{}^2}), \ldots, (x_{c_1,}y_{c_1})\}$,

$r_{(s,d)}{}^{2}{}_{2} = \{(x_{c_1,}y_{c_1}), (x_{n\_{}^2,}y_{n\_{}^2}), \ldots, (x_{c_2,}y_{c_2})\}$,

$r_{(s,d)}{}^{2}{}_{i} = \{(x_{c_{i-1},}y_{c_{i-1}}), (x_{n\_{}^2,}y_{n\_{}^2}), \ldots, (x_{c_i,}y_{c_i})\}$,

$r_{(s,d)}{}^{2}{}_{m+1} = \{(x_{c_m,}y_{c_m}), \ldots, (x_{n_{h_2-1}{}^2,}y_{n_{h_2-1}{}^2}), (x_{d,}y_d)\}$

Step 3: Merge to get the following two children –

$$r'_{(s,d)}{}^{1} = r_{(s,d)}{}^{1}{}_{1} \parallel r_{(s,d)}{}^{2}{}_{2} \parallel r_{(s,d)}{}^{1}{}_{3} \parallel r_{(s,d)}{}^{2}{}_{4} \parallel \cdots \parallel r_{(s,d)}{}^{1}{}_{m} \parallel r_{(s,d)}{}^{2}{}_{m+1}$$

$$r'_{(s,d)}{}^{2} = r_{(s,d)}{}^{2}{}_{1} \parallel r_{(s,d)}{}^{1}{}_{2} \parallel r_{(s,d)}{}^{2}{}_{3} \parallel r_{(s,d)}{}^{1}{}_{4} \parallel \cdots \parallel r_{(s,d)}{}^{2}{}_{m} \parallel r_{(s,d)}{}^{1}{}_{m+1}$$

when *m* is odd, or

$$r'_{(s,d)}{}^{1} = r_{(s,d)}{}^{1}{}_{1} \parallel r_{(s,d)}{}^{2}{}_{2} \parallel r_{(s,d)}{}^{1}{}_{3} \parallel r_{(s,d)}{}^{2}{}_{4} \parallel \cdots \parallel r_{(s,d)}{}^{2}{}_{m} \parallel r_{(s,d)}{}^{1}{}_{m+1}$$

$$r'_{(s,d)}{}^{2} = r_{(s,d)}{}^{2}{}_{1} \parallel r_{(s,d)}{}^{1}{}_{2} \parallel r_{(s,d)}{}^{2}{}_{3} \parallel r_{(s,d)}{}^{1}{}_{4} \parallel \cdots \parallel r_{(s,d)}{}^{1}{}_{m} \parallel r_{(s,d)}{}^{2}{}_{m+1}$$

when *m* is even, and || is the merging operator.

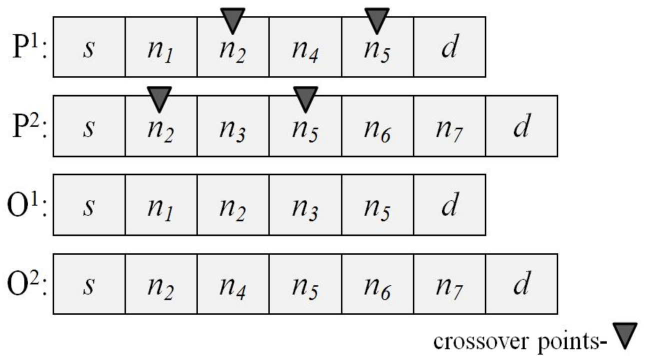


Figure 2 Example of multi-point crossover

Fig. 2 shows the alternating exchange of partial paths from parents (p^1, p^2) to produce offspring chromosomes (O^1, O^2) in a 2-point crossover.

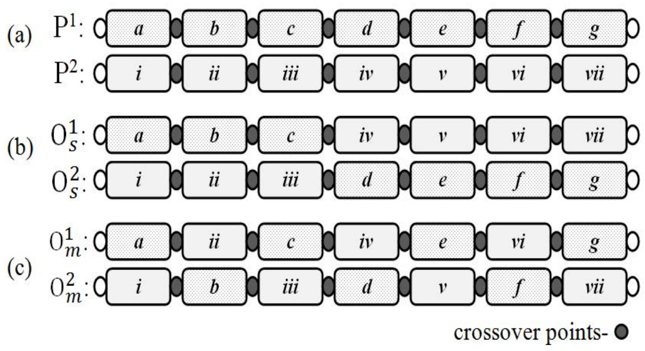


Figure 3. (a) Parent chromosomes, (b) Offspring produced by single-point crossover, (c) Offspring produced by multi-point crossover

The multi-point crossover is more advantageous in creating diversity, particularly in case of relatively long chromosomes. This is evident in Fig. 3; the offspring produced by single-point crossover have long common sub-sequences with their parents, whereas the offspring in multi-point crossover differ largely from their parents. On the flipside, it adds to computational complexity. In Fig. 3, each block is a distinct partial path, i.e. a distinct sequence of genes. However, they may have common genes. This may induce multiple loops after crossover.

F. Mutation

The mutation operation introduces a small random bias in the population in the hope of recovering lost genes and driving the convergence away from local optima. Mutation operator is applied only on a fraction of the total population. Let $r_{(s,d)} = \{s, n_1, n_2, \dots, n_{h-1}, d\}$ be a candidate chromosome for mutation. All genes except the source *s* and destination *d* cells are mutable. Mutation operation proceeds as follows:

```
Step 1: Randomly select mutation point $n_m$, $1 \le m < h$.
Step 2: Remove all genes with locus > $m$.
Step 3: Randomly select a path $p$ from $P_{(n_m,d)}$.
Step 4: Append $p$ to $r$.
```

Mutation generates an alternate partial path from the mutation point to the destination cell. This can again result in loops when the alternate partial path and the surviving portion of the original path have gene(s) in common. Fig. 4 shows an example mutation operation.

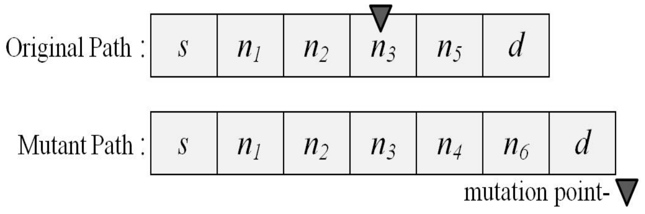


Figure 4. Example of mutation operation

G. Repair function

Crossover and mutation may produce infeasible chromosomes with loops in the routing path. The repair function finds and eliminates all loops. Multi-point crossover and mutation can result in multiple occurrences of the following three basic types of loops as shown in Fig 5.

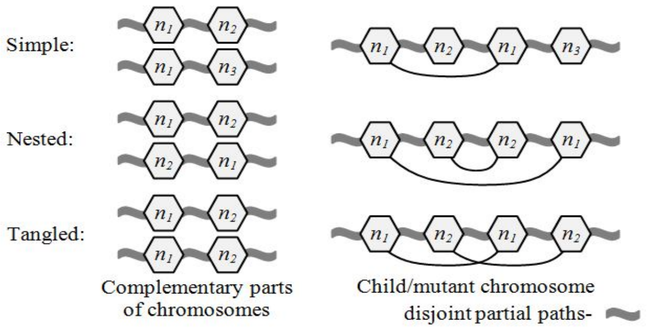


Figure 5. Possible loop formations

Loops are repaired by deleting the genes between the farthest repeated genes and one occurrence of the duplicate gene. A defective chromosome $r_{(s,d)}$ of length *h* is repaired as follows:

```
Step 1: For $i = 0$ to $h$, repeat step 2.
Step 2: For $j = 0$ to $h-i-1$, repeat step 3.
Step 3: If $n_i = n_{h-j}$, then
            $r_{(s,d)} = \{n_0, n_1, \dots, n_i\} \parallel \{n_{h-j+1}, \dots, n_h\}$,
            $h = i + j$.
```

Fig. 6 shows how the repair function tackles different types of loops. The function runs in O(h^2) time.

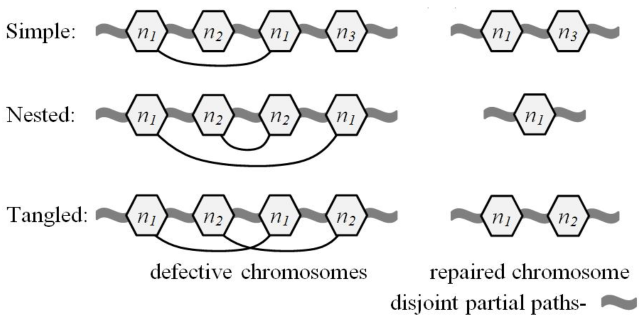
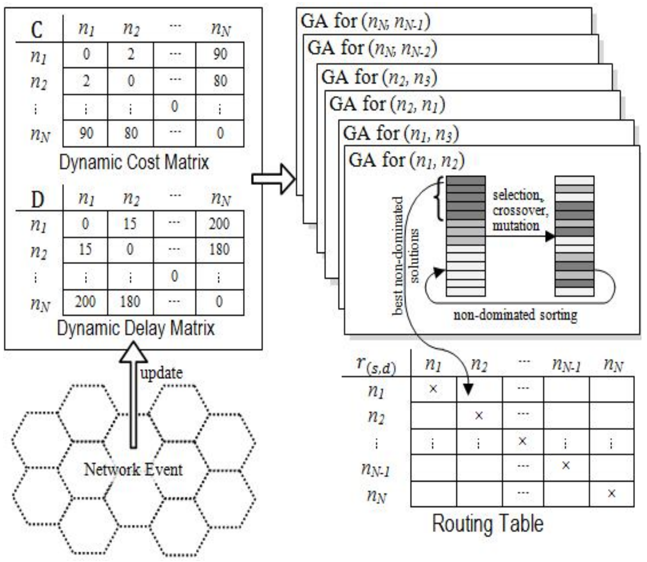


Figure 6. Repairing loops

III. Real Time Framework

The SP routing problem for each source-destination pair is formulated into a multi-objective GA. The overall complexity of the NSGA-II algorithm is . *O(MN²)*, where *M* is the number of objectives and *N* is the population size. Moreover, the GAs require several iterations to converge to the pareto-optimal solutions. In reality, the traffic demand, congestion and network status changes continuously. The corresponding changes in the cost and delay matrices require the GAs to be restarted or reiterated. As a result, the response times in such dynamic environment are unacceptably high. This challenges the real-time implementation of GAs for routing.



C	n_1	n_2	···	n_N
n_1	0	2	···	90
n_2	2	0	···	80
⋮	⋮	⋮	0	⋮
n_N	90	80	···	0

D	n_1	n_2	···	n_N
n_1	0	15	···	200
n_2	15	0	···	180
⋮	⋮	⋮	0	⋮
n_N	200	180	···	0

$r_{(s,d)}$	n_1	n_2	···	n_{N-1}	n_N
n_1	×		···		
n_2		×	···		
⋮	⋮	⋮	×	⋮	⋮
n_{N-1}			···	×	
n_N			···		×

Figure 7. Real-time framework

We propose a parallel, distributed and reactive framework (Fig. 7) for real-time implementation of GAs for dynamic routing in mobile networks. Each base station runs several GAs with itself as the source. To save computational resources, only the GAs for highly active cell pairs are initialized. A call request between a new source-destination pair triggers the corresponding GA on-demand. The cost and delay matrices update dynamically in response to network events like congestion or component failure. After each iteration of a GA, the set of best paths (first non-domination front) among the population is copied into the corresponding routing table entry. An incoming call request is routed instantly by consulting the routing table without waiting for the GA to converge.

IV. Simulation Results

To evaluate the performance of the proposed GA, simulations were performed with MATLAB 7.10. GAs were executed for random source-destination pairs in a 10×10 cellular field with 100 cells. The population size and maximum number of generations were selected as 20 and 50 respectively. The crossover and mutation probabilities were set to $P_c = 1$ and $P_m = 0.2$ respectively. The cost and delay values were assigned in the range [1,100] and [10,300] respectively.

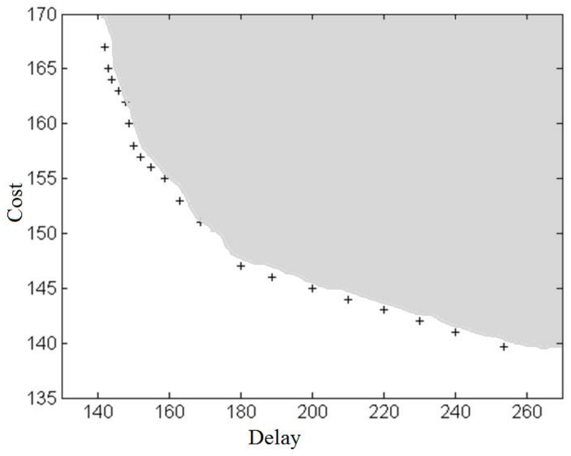


Figure 8. Pareto-optimal front in last generation

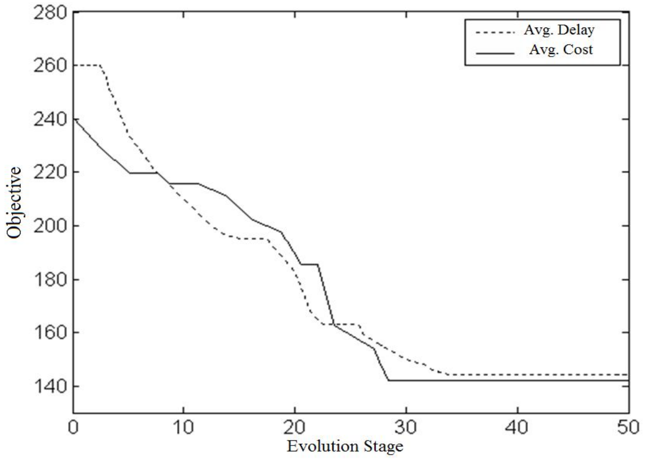


Figure 9. Convergence of cost and delay objectives

Fig. 8 shows a pareto-optimal front obtained by NSGA-II. The shaded region is the feasible solution space and the dashed curve specifies the true pareto-optimal solutions. A large number of solutions (+) are found in a single run. Fig. 9 shows the convergence of cost and delay objectives through the evolution stages; the curves plot the average cost and delay of the current non-dominated front.

V. Conclusions

This paper formulates the dynamic routing problem in mobile networks as a multi-objective optimization problem and uses the NSGA-II algorithm to solve it. An improved crossover operator has also been defined and investigated. Simulations show that NSGA-II is efficient for multi-objective routing and results in a large number of pareto-optimal solutions, and a good spread among them, in a single run. Interestingly, the proposed real-time framework is a promising direction and invites further research.

References

[1] R. Poddar, S. Banerjee, and P. K. Guha Thakurta, “Shortest Path Routing for Co-ordinate based Mobile Networks: A Genetic Algorithm Approach”, In proc. International Conference on Advances in Mobile Network, Communication and its Applications, MNCApps, Bangalore, August 2012.

[2] S. Banerjee, S. Roy, and P. K. Guha Thakurta, “Coordinate based Directed Routing Protocol”, International Journal of Information and Electronics Engineering, vol. 2(2):170-173, March 2012.

[3] Chang Wook Ahn, and R.S. Ramakrishna, “A genetic algorithm for shortest path routing problem and the sizing of populations”, in IEEE Transactions on Evolutionary Computation, vol. 6. no .6., pp. 566 - 579, December 2002.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II”, in IEEE Transactions on Evolutionary Computation, vol. 6. no. 2, pp. 182 - 197, April 2002.

[5] T. Back, “Selective Pressure in Evolutionary Algorithms: A Characterization of Selection Mechanisms”, First IEEE Conference on Evolutionary Computaion: IEEE World Congress on Computational Intelligence, pp. 57-62, 27-29 June 1994.

[6] D. E. Goldberg, K. Deb, and J. H. Clark, “Genetic algorithms, noise, and the sizing of populations”, Complex Syst., vol. 6, pp. 333–362, 1992.

[7] W. Stalling, High-Speed Networks: TCP/IP and ATM Design Principles. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[8] G. H. Golub, Charles F. Van Loan, Matrix Computations (3rd ed.), Englewood Cliffs, NJ: Prentice-Hall, 1998; Baltimore: Johns Hopkins university Press, 1996.